

Generator parsera

CUP v11a

Uvod

- Alat CUP generiše java kod LALR(1) parsera na osnovu specifikacije u vidu bezkontekstne atributivno-translacione gramatike

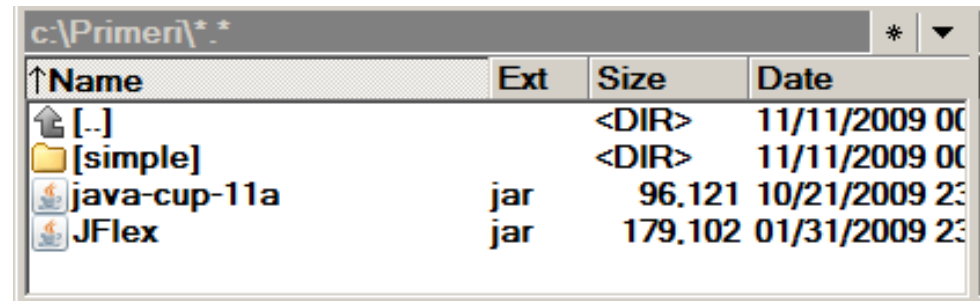
Primer jednostavnog evaluatora aritmetičkih izraza (paket simple)

Funkcionalni zahtevi:

- Sa standardnog ulaza dobija niz (razdvojen ;) aritmetičkih izraza (operatori +, -, *, /, % i zagrade) sa celobrojnim konstantama:
 - $12+3*(5-1); 23;$
- Na izlazu ispisuje vrednosti svakog od izraza:
 - $=24$
 - $=23$
- Kao oznaka kraja ulaza se zadaje ctrl-Z

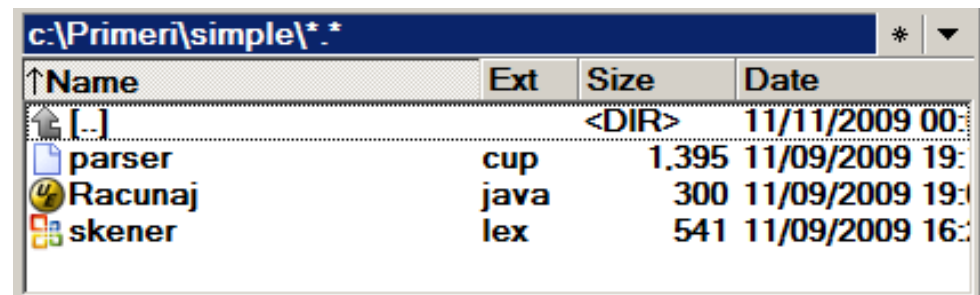
Struktura projekta simple

- U korenu se nalaze alati **java-cup-11a.jar** i **JFlex.jar**
- U folderu **simple** nalaze se specifikacija skenera **skener.lex**, specifikacija parsera **parser.cup** i glavni program **Racunaj.java**



Windows Explorer window showing the root directory `c:\Primer*.*`. The table lists the following files and directories:

Name	Ext	Size	Date
[..]	<DIR>		11/11/2009 00:...
[simple]	<DIR>		11/11/2009 00:...
java-cup-11a	jar	96,121	10/21/2009 23:...
JFlex	jar	179,102	01/31/2009 23:...



Windows Explorer window showing the contents of the `c:\Primer\simple*.*` directory. The table lists the following files:

Name	Ext	Size	Date
[..]	<DIR>		11/11/2009 00:...
parser	cup	1,395	11/09/2009 19:...
Racunaj	java	300	11/09/2009 19:...
skener	lex	541	11/09/2009 16:...

skener.lex

- Ovaj fajl predstavlja specifikaciju skenera koji se automatski generiše alatom jflex

```
package simple;  
import java_cup.runtime.Symbol;
```

```
%%
```

```
%cup
```

```
%eofval{  
    return new Symbol(sym.EOF);  
}%eofval}
```

```
%%
```

```
" " {}
```

```
\b {}
```

```
\t {}
```

```
\r\n {}
```

```
\f {}
```

```
"+" { return new Symbol(sym.PLUS); }
```

```
"-" { return new Symbol(sym.MINUS); }
```

```
"/" { return new Symbol(sym.DIVIDE); }
```

```
"%" { return new Symbol(sym.MOD); }
```

```
"*" { return new Symbol(sym.TIMES); }
```

```
"(" { return new Symbol(sym.LPAREN); }
```

```
")" { return new Symbol(sym.RPAREN); }
```

```
 ";" { return new Symbol(sym.SEMI); }
```

```
[0-9]+ { return new Symbol(sym.NUMBER, new Integer(yytext()) ); }
```

```
. {}
```

skener.lex

```
package simple;  
import java_cup.runtime.Symbol;  
  
%%
```

- Sadržaj prve sekcije direktno se preslikava u izlazni java fajl. Import deklaracija je neophodna jer je klasa simbol deo standardnog cup interfejsa između skenera i parsera. Token se prosleđuje parseru kao nova instanca klase Symbol

skener.lex

```
%%
```

```
%cup
```

```
%eofval{  
    return new Symbol(sym.EOF);  
}%eofval}
```

```
%%
```

- Sadržaj druge sekcije utiče na izgled generisanog java koda. %cup nalaže da skener ima cup interfejs prema parseru
- Eofval nalaže da se na kraju ulaza parseru vrati token EOF. Ova konstanta definisana je u klasi sym koju cup alat automatski generiše iz gramatike (svaki terminal dobija svoju konstantu, plus EOF i error).

Standardni cup interfejs skenera prema parseru:

- Skener mora implementirati interfejs **java_cup.runtime.Scanner**
- Funkcija tog interfejsa koju parser poziv za dobijanje sledećeg terminala je **next_token()**
- Povratna vrednost ove funkcije je nova instanca klase **java_cup.runtime.Symbol**

skener.lex

```
%%  
  
" " {}  
\b {}  
\t {}  
\r\n {}  
\f {}  
"+" { return new Symbol(sym.PLUS); }  
"- " { return new Symbol(sym.MINUS); }  
"/" { return new Symbol(sym.DIVIDE); }  
"% " { return new Symbol(sym.MOD); }  
"*" { return new Symbol(sym.TIMES); }  
"(" { return new Symbol(sym.LPAREN); }  
")" { return new Symbol(sym.RPAREN); }  
";" { return new Symbol(sym.SEMI); }  
[0-9]+ { return new Symbol(sym.NUMBER, new Integer(yytext())); }  
. {}
```

- Treća lex sekcija definiše regularne izraze i akcije.
- Kada token ima samo klasni deo a ne i vrednosni (tj. Terminal u gramatici nema atribut) onda se koristi konstruktor klase Symbol sa jednim celobrojnim argumentom.
- Ako token ima vrednosni deo (npr. NUMBER) onda dva argumenta, drugi je tipa reference na klasu Object, treba proslediti onu klasu koja je deklarirana u Cup fajlu za taj terminal
- Poslednji reg.izraz je catch-all tipa da se “pojede” nelegalni znak na ulazu

CUP specifikacija

Opšta struktura .cup fajla

- package i import deklaracije (neobavezno),
- deklaracije korisničkog koda (neobavezno),
- deklaracije terminalnih i neterminalnih simbola,
- precedence deklaracije (neobavezno)
- gramatika

Primer: parser.cup

- .cup
specifikacija za
projekat
jednostavnog
interpretera
izraza

```
package simple;
// CUP specification for a simple expression evaluator (w/ actions)

import java_cup.runtime.*;

/* Terminals (tokens returned by the scanner). */
terminal      SEMI, PLUS, MINUS, TIMES, DIVIDE, MOD;
terminal      LPAREN, RPAREN;
terminal Integer  NUMBER;

/* Non-terminals */
non terminal   expr_list, expr_part;
non terminal Integer  expr;

/* Precedences */
precedence left PLUS, MINUS;
precedence left TIMES, DIVIDE, MOD;

/* The grammar */
expr_list ::= expr_list expr_part
          |
          expr_part;

expr_part ::= expr:e
          { : System.out.println("= " + e); :}
          SEMI
          ;

expr ::= expr:e1 PLUS expr:e2
      { : RESULT = new Integer(e1.intValue() + e2.intValue()); :}
      |
      expr:e1 MINUS expr:e2
      { : RESULT = new Integer(e1.intValue() - e2.intValue()); :}
      |
      expr:e1 TIMES expr:e2
      { : RESULT = new Integer(e1.intValue() * e2.intValue()); :}
      |
      expr:e1 DIVIDE expr:e2
      { : RESULT = new Integer(e1.intValue() / e2.intValue()); :}
      |
      expr:e1 MOD expr:e2
      { : RESULT = new Integer(e1.intValue() % e2.intValue()); :}
      |
      NUMBER:n
      { : RESULT = n; :}
      |
      LPAREN expr:e RPAREN
      { : RESULT = e; :}
      ;
```

Package i import deklaracije

```
package simple;  
  
import java_cup.runtime.*;
```

- Preslikavaju se direktno u generisani java fajl

Korisnički kod (opciona sekcija pre terminalnih i neterminalnih simbola)

- **action code** { : /* java code */ : }
 - java code1 vidljiv je u akcijama (smešta se u parser actions klasu)
- **parser code** { : /* java code */ : }
 - Smešta se u parser klasu
 - Služi na primer da se u klasu parser ubaci metod main() za pokretanje parsera sa komandne linije – videti primer PPDZ
- **init with** { : /* java code */ : }
 - Izvršava se pre čitanja prvog tokena (simbola)
 - Moguća upotreba: kreiranje instance skenera
- **scan with** { : /* java code */ : }
 - Izvršava parser kada mu treba naredni simbol
 - Služi npr. da se izmeni standardni poziv skenera od strane parsera

Deklaracija terminalnih simbola

terminal SEMI, PLUS, MINUS, TIMES, DIVIDE, MOD;
terminal LPAREN, RPAREN;
terminal **Integer** NUMBER;

- Predstavljaju tokene iz skenera
+ - * / % ; () NUMBER
- Za svaki terminal CUP generiše jednu konstantu u klasi sym.java
- Neophodno ih je deklarirati da bi ih CUP razlikovao od neterminala i da bi se definisao tip eventualnog atributa (vidi NUMBER)
- Za razliku od originalnog yacc alata ne dozvoljavaju se imena terminala u vidu znakovnih konstanti ‘;’ i slično

Deklaracija neterminala

```
non terminal          expr_list, expr_part;  
non terminal Integer  expr;
```

- Takođe je obavezno deklarirati sve gramatičke neterminale (i tip atributa ako ga poseduju)
- Sledeće reči ne smeju se koristiti za nazive term. i neterm: code, action, parser, terminal, non, nonterminal, init, scan, with, start, precedence, left, right, nonassoc, import i package
- Postoji jedan startni neterminal

Smene

```
/* The grammar */  
expr_list ::= expr_list expr_part  
           | expr_part  
           ;  
  
expr_part ::= expr  
           SEMI  
           ;
```

- ::= služi umesto →
- Smena može i u više redova, blanko se ignoriše
- | služi kao oznaka da je leva strana ista kao u prethodnoj smeni. ; završava zapis niza smena sa istom levom stranom
- Opciono pre gramatike može delaracija startnog neterminala **start with** *neterminal*; Ako se ne navede, podrazumeva se leva strana prve smene

Atributi (labele) i akcioni kod

```
expr_part ::= expr:e
           {: System.out.println("=" + e); :}
           SEMI
           ;
expr      ::= expr:e1 PLUS expr:e2
           {: RESULT = new Integer(e1.intValue() + e2.intValue()); :}
```

- Java kod koji se izvršava u smenama, ograničava se sa {: i :}
- Izvršavaju se kada se smena redukuje. Ako je u sredini smene, cup automatski uvodi novi neterminal na mestu akcije i praznu smenu za njega koja aktivira akciju (poljska transl. gramatika) – pažljivo koristiti, može da izazove parserske konflikte
- Atributi se imenuju sa :ime. Oni su tipa koji je naveden u deklaraciji. Atribut neterminala leve strane uvek se zove RESULT

Korišćenje parsera (glavni program – projekat simple)

```
package simple;

class Racunaj {
    public static void main(String args[]) throws Exception {
        /* create a parsing object */
        parser parser_obj = new parser( new Yylex(System.in));
        parser_obj.parse(); // ili debug_parse() ako hocemo ispis
                           // parserskih akcija na System.err
    }
}
```

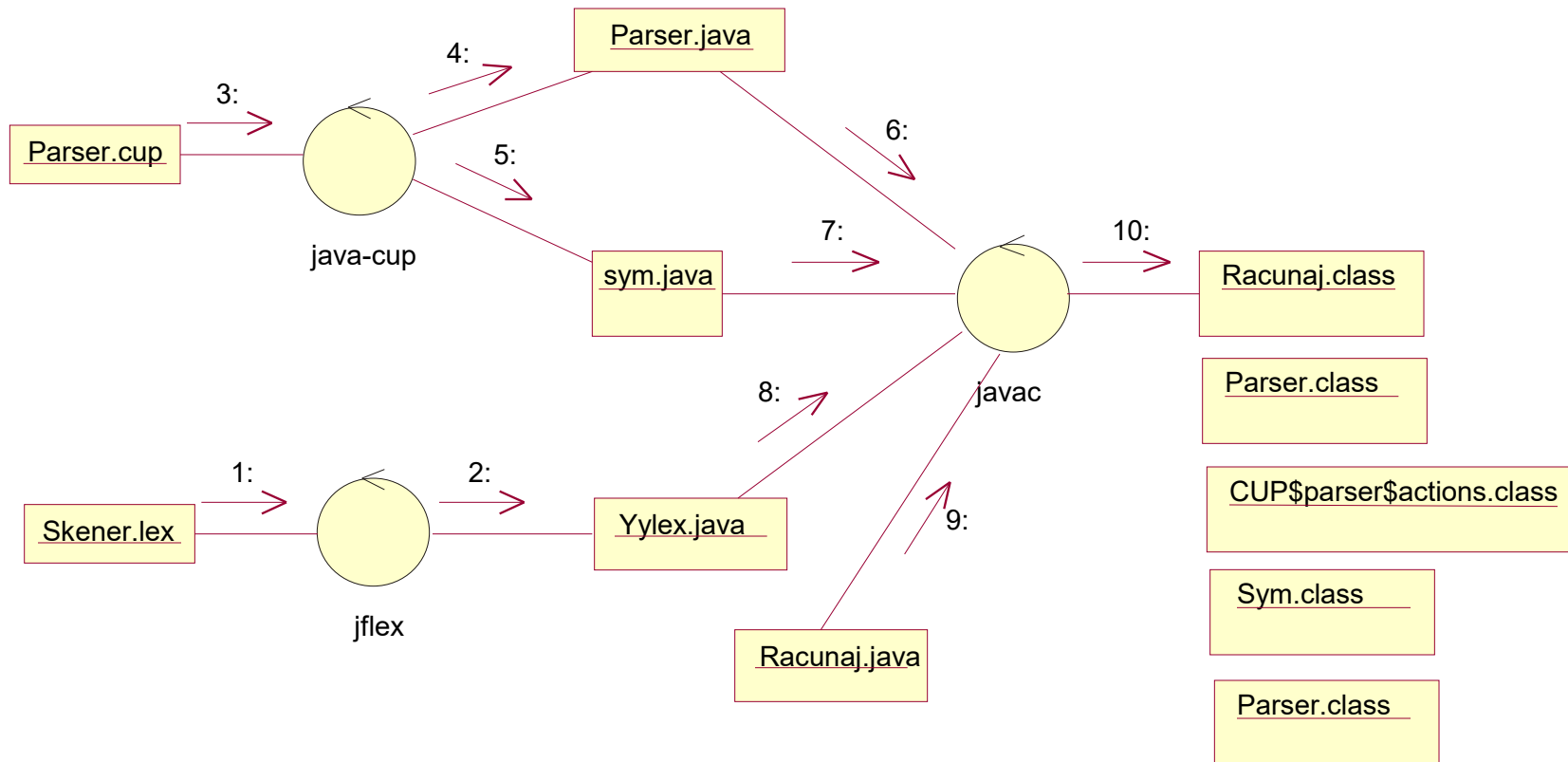
• kreira se instanca klase parser i u konstruktoru dostavlja nova instanca skenera. Skener čita sa standardnog ulaza. Druga varijanta je:

- **parser parser_obj = new parser();**
- **parser_obj.setScanner(new YYLex(...));**

Korišćenje parsera (glavni program – projekat simple)

- Napomene:
- `main()` je deklarisan sa `throws Exception` jer `parse()` može baciti izuzetak pri sintaksoj grešci na ulazu (a nismo predvideli nikakvu drugu obradu greške)
- Poziv funkcije `parse()` izaziva kompletno parsiranje ulaza. Ako startni neterminal ima atribut tipa `T`, onda `parse()` vraća objekat `Symbol` čije `value` polje sadrži objekat tipa `T`, vrednost atributa startnog simbola
- `debug_parse()` radi što i `parse()` i dodatno na `System.err` šalje detaljan prikaz rada parsera po koracima (može se preusmeriti u fajl sa `2>koraci.txt`):

Pokretanje alata i prevođenje programa



Pokretanje alata i prevođenje programa – projekat simple

- Preduslov je da sistemska PATH promenljiva pokazuje na bin folder java jdk instalacije.
- Komande se zadaju u Command Promptu. Tekući direktorijum je koren primera, gde se nalazi jar arhiva cup alata:
- **java -jar JFlex.jar simple\skener.lex**
(generiše skenersku klasu Ylex.java u folderu simple)
- **java -jar java-cup-11a.jar -destdir simple simple\parser.cup**
(generiše parser.java i sym.java u folderu simple)
- **javac -cp .;java-cup-11a.jar simple\Racunaj.java**
(prevodi Racunaj.java, parser.java, Ylex.java i sym.java i generiše odgovarajuće .class fajlove)
- **java -cp .;java-cup-11a.jar simple.Racunaj**
(pokreće program)

Pokretanje alata i prevođenje programa – projekat simple

- Korisna opcija pri pozivu CUP-a je `–dump_states`, ona daje karakteristični LALR(1) automat sa konfiguracijama, npr:

```
-----  
lalr_state [6]: {  
  [expr ::= expr (*) PLUS expr , {PLUS MINUS TIMES DIVIDE MOD RPAREN }]  
  [expr ::= expr (*) TIMES expr , {PLUS MINUS TIMES DIVIDE MOD RPAREN }]  
  [expr ::= expr (*) MOD expr , {PLUS MINUS TIMES DIVIDE MOD RPAREN }]  
  [expr ::= expr (*) MINUS expr , {PLUS MINUS TIMES DIVIDE MOD RPAREN }]  
  [expr ::= LPAREN expr (*) RPAREN , {SEMI PLUS MINUS TIMES DIVIDE MOD RPAREN }]  
  [expr ::= expr (*) DIVIDE expr , {PLUS MINUS TIMES DIVIDE MOD RPAREN }]  
}  
transition on TIMES to state [12]  
transition on DIVIDE to state [11]  
transition on MINUS to state [10]  
transition on PLUS to state [9]  
transition on RPAREN to state [8]  
transition on MOD to state [7]  
-----
```

- Ovo se može koristiti pri razrešavanju S/R i R/R konflikata

Deklaracije prioriteta i asocijativnosti operatora

- Ako u projektu simple iz fajla parser.cup izbacimo linije:

precedence left PLUS, MINUS;

precedence left TIMES, DIVIDE, MOD;

- CUP će pri generisanju parsera javiti sledeću niz grešaka (sledeći slajd) i neće izgenerisati parser

Warning : *** Shift/Reduce conflict found in state #16
between $\text{expr} ::= \text{expr PLUS expr } (*)$
and $\text{expr} ::= \text{expr } (*) \text{ PLUS expr}$
under symbol PLUS
Resolved in favor of shifting.

Warning : *** Shift/Reduce conflict found in state #13
between $\text{expr} ::= \text{expr TIMES expr } (*)$
and $\text{expr} ::= \text{expr } (*) \text{ MOD expr}$
under symbol MOD
Resolved in favor of shifting.

Error : *** More conflicts encountered than expected -- parser generation aborted

----- CUP v0.11a beta 20060608 Parser Generation Summary -----

1 error and 25 warnings

11 terminals, 4 non-terminals, and 12 productions declared,
producing 22 unique parse states.

0 terminals declared but not used.

0 non-terminals declared but not used.

0 productions never reduced.

25 conflicts detected (0 expected).

No code produced.

----- (v0.11a beta 20060608)

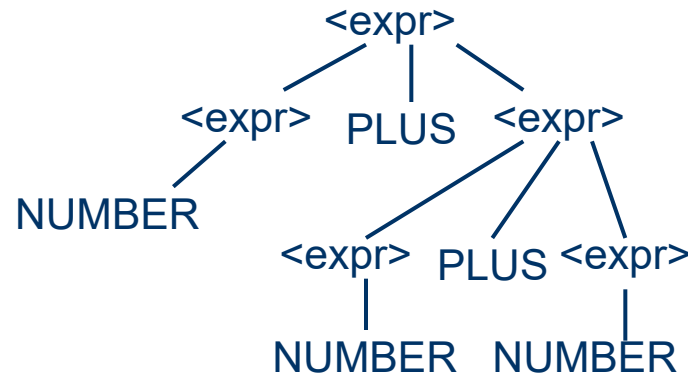
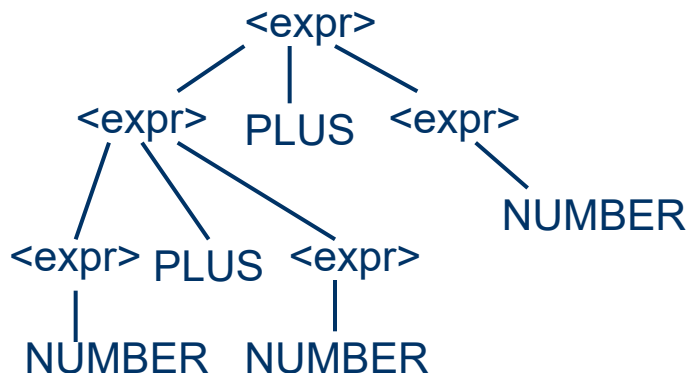
Deklaracije prioriteta i asocijativnosti operatora

- Razlog je dvosmislenost zadate gramatike u delu `expr`:

`expr ::= expr PLUS expr`

`.....`
`| NUMBER`

- Dva različita stabla izvođenja za
`NUMBER PLUS NUMBER PLUS NUMBER`



Deklaracije prioriteta i asocijativnosti operatora

- Deo karakterističnog automata (prikazane samo konfiguracije od interesa):

$\langle \text{Expr} \rangle ::= \langle \text{Expr} \rangle \text{ PLUS } \bullet \langle \text{Expr} \rangle, \{\text{PLUS}\}$
 $\langle \text{Expr} \rangle ::= \bullet \langle \text{Expr} \rangle \text{ PLUS } \langle \text{Expr} \rangle$
 $\langle \text{Expr} \rangle ::= \bullet \text{NUMBER}$

$\langle \text{Expr} \rangle$

Stanje #16

$\langle \text{Expr} \rangle ::= \langle \text{Expr} \rangle \text{ PLUS } \langle \text{Expr} \rangle \bullet, \{\text{PLUS}\}$
 $\langle \text{Expr} \rangle ::= \langle \text{Expr} \rangle \bullet \text{ PLUS } \langle \text{Expr} \rangle$

- U stanju #16 nastaje SHIFT/REDUCE konflikt za ulazni simbol PLUS

Deklaracije prioriteta i asocijativnosti operatora

- CUP razrešava konflikte (ako se ništa ne zada) tako da S/R uvek razreši u korist SH, a kod R/R konflikta u korist smene koja je prva navedena u .CUP fajlu. Ovo nije uvek ono što mi želimo.
- Možemo da forsiramo generisanje parsera sa razrešenim konfliktima ako pri pozivu CUPa dodamo **–expect 25**

----- CUP v0.11a beta 20060608 Parser Generation Summary -----

0 errors and 25 warnings

11 terminals, 4 non-terminals, and 12 productions declared,
producing 22 unique parse states.

0 terminals declared but not used.

0 non-terminals declared but not used.

0 productions never reduced.

25 conflicts detected (25 expected).

Code written to "parser.java", and "sym.java".

----- (v0.11a beta 20060608)

Deklaracije prioriteta i asocijativnosti operatora

- Deklaracije:
precedence left PLUS, MINUS;
precedence left TIMES, DIVIDE, MOD;
- Utiču na to kako će parser razrešiti konflikte. Ovo znači da su + i – istog prioriteta i levo asocijativne, a * / % međusobno istog prioriteta, višeg od + i – i takođe levo asocijativne

Kako ovo utiče na razrešavanje konflikata?

- Smena dobija prioritet prema poslednjem terminalu na njenoj desnoj strani, npr. **<Expr> ::= <Expr> PLUS <Expr>** ima prioritet kao **PLUS** i **MINUS**
- U konfliktnom stanju poredi se prioritet relevante smene sa prioritetom tekućeg ulaznog simbola. Ako je prioritet smene viši, npr. **<Expr> ::= <Expr> TIMES <Expr>** a ulazni simbol je **PLUS**, onda se bira REDUCE u suprotnom se bira SHIFT
- Ako su prioriteti isti, gleda se asocijativnost. Ako je leva asocijativnost, bira se REDUCE smene, ako je desna asocijativnost bira se SHIFT akcija. Ako se za asocijativnost navede **nonassoc**, parser će u ovoj situaciji prijaviti sintaksnu grešku.

Deklaracije prioriteta i asocijativnosti operatora

- Terminali koji se ne pojave u **precedence** deklaraciji imaju najniži prioritet.
- Konačno, smeni se u gramatici može eksplicitno dodeliti prioritet **%prec** deklaracijom, na primer:
- $\langle X \rangle ::= Y Z \{ : \text{neka akcija} : \} \% \text{prec } T$
- U ovom primeru, smeni za $\langle X \rangle$ dodeljuje se prioritet terminala T

Sintaksne greške

- Ako u primeru simple zadamo ulaz: #1; evaluator izraza će ignorisati grešku zbog catch-all izraza u skeneru koji će 'pojesti' #
- Međutim, ako zadamo 1+; skener radi normalno, a grešku detektuje parser i baca izuzetak:

```
C:\Primeri>java -cp .;java-cup-11a.jar simple.Racunaj  
1+;
```

```
Syntax error
```

```
Couldn't repair and continue parse
```

```
Exception in thread "main" java.lang.Exception: Can't recover from previous error(s)  
    at java_cup.runtime.lr_parser.report_fatal_error(lr_parser.java:375)  
    at java_cup.runtime.lr_parser.unrecovered_syntax_error(lr_parser.java:424)  
    at java_cup.runtime.lr_parser.parse(lr_parser.java:616)  
    at simple.Racunaj.main(Racunaj.java:7)
```

Sintaksne greške

- U trenutku detekcije greške (a pre pokušaja oporavka ako postoje error smene), parser poziva svoj metod:
`public void syntax_error(Symbol cur_token)`
koji, ako nije preinačen, poziva `report_error("syntax error")` za ispis na `System.err`
- Ako nema error smena, ili oporavak nije uspeo, poziva se metod
`public void unrecovered_syntax_error(Symbol cur_token)`
koji u originalu poziva `report_fatal_error("Couldn't repair and continue parse")` koja ispisuje grešku, poziva `done_parsing()` i konačno baca izuzetak.
- Za primer kako se upotrebom **parser code** deklaracije u cup fajlu mogu izmeniti ove funkcije pogledati projekat **ppdz**.

Oporavak od grešaka

- Predefinisan neterminal error, ne pojavljuje se na ulazu u normalnom režimu rada
- Normalnim smenama se dodaju posebne error smene, na primer za projekat simple:

`expr_part ::= expr:e`

`SEMI { : System.out.println("= " + e); : }`

`| error`

`;`

Oporavak od grešaka

```
expr_part ::= expr:e  
SEMI {: System.out.println("= " + e); :}  
| error  
;
```

- Ovime se obezbeđuje oporavak i za slučaj grešaka u izrazu i za slučaj izostavljanja ;
- Parser sada javlja “syntax error”, ali nastavlja rad i ne baca izuzetak

```
C:\Primeri>java -cp .;java-cup-11a.jar simple.Racunaj  
1+;2;  
Syntax error  
^Z  
= 2
```

```
C:\Primeri>java -cp .;java-cup-11a.jar simple.Racunaj  
1 2;3;  
Syntax error  
= 2  
^Z  
= 3
```